ABIBA

ABIBA: An Agent-based Computing System for Behaviour Analysis used in Human-Agent Interaction

Can Cui[1], Dave Murray-Rust[2], David Robertson[1], and Kristin Nicodemus[3]

[1] School of Informatics, The University of Edinburgh, Edinburgh, United Kingdom
`c.cui@ed.ac.uk, dr@inf.ed.ac.uk`
[2] School of Design, The University of Edinburgh, Edinburgh, United Kingdom
[3] Institute of Genetics and Molecular Medicine, The University of Edinburgh, Edinburgh, United Kingdom

Abstract. We build an agent-based system for supporting correlation analysis between human behavioural and non-behavioural patterns. A novel social norm specification language is leveraged to create an interaction model based communication engine for choreographing distributed systems, offering a communication environment for multiple interacting players. Categorising sets of players based on their interaction behaviours allows labelling the other patterns, which the system uses to further its understanding relationship between the two traits. While existing analysis methods are manually applied, non-user-editable and typically opaque, the system offers an end-to-end computing framework and protocols which are modifiable for specific users. Evaluation for this system relies on tests for categories of people who are mentally depressed, where traditional questionnaire-based methods are superseded by methods that use more objective behavioural tests. This approach to evaluation through behavioural experimentation is intended not only to classify sub-types of depression cases which would facilitate elucidation of aetiology but evaluates system performance in a real-world scenario.

Keywords: multiagent system, social norm, interaction simulation, behaviour analysis, human-agent interaction, computational psychiatry

1 Introduction

In this work, we leverage social norms to propose a novel agent-based framework for human-agent interaction analysis. Social norms are the customary rules that govern individuals' social behaviours. The concept of social norms is suitable for building agent-based system because the agents have to follow basic rules of interactions to complete certain cooperate tasks. We use the Lightweight Coordination Calculus (LCC) introduced in [7] to specify the social norms. The multi-agent systems traditionally use electronic institutions (and other forms of the executable social norm) to ensure that the behaviours of each agent stay within the confines of the appropriate social norm. In this work, by contrast,

we have the principal (additional) purpose that we use electronic institutions to elicit behaviours from agents that can then be used to cluster and classify them into different sets. For this purpose, we build the Agent-based Interaction Behaviour Analysis System (ABIBA) which can manage multi-agent interactions analysis. The system is an end-to-end solution for the system users who plan analyse agents behavioural patterns through multi-agent interactions. The users can obtain the analysis results without mastering many topic unrelated engineering skills for building the interaction and collection platform. Their only job is to make experimental protocols which the system will follow to create agents and hold the interactions. Regarding agents' patterns analysis, the users can choose the system to do specific predesigned tasks or build the ones themselves from results from the previous interactions.
To show how reasoning for the behavioural analysis can be automated and, as proof of concept, we have started a case study in which our system is used to support the study about understanding the aetiology of Major Depressive Disorder (MDD). MDD is a clinically significant degree of depression that is highly prevalent in the population [12]. It's vital for the society to understand MDD further. Even though many works have made valuable contributions to the task, the traditional research methods are time-consuming for researchers to ascertain participants in the task experiment and analyse the data from various domains. The ABIBA system will help researchers more efficiently manage behavioural research and develop data analysis.
The remainder of the paper is structured as follow: After reviewing the related work in section 2, we present details about ABIBA system in section 3. In section 4 we explain the behaviour experiments focused on MDD including experimental instruments, experimental protocols and behavioural analysis.

2 Related Work

Most existing works on normative multi-agent system area focus on building framework demonstrating agents to behave under social norms while the purpose of our work is to analyse the emergent behaviours of agents when they interact with each other. [3] uses model checking approaches to verify agents' behaviours against predefined models, [1] concentrates on evaluating BDI agent design against the requirements. These approaches typically concentrated on testing, debugging and verification of multi-agent system [9] which only verify the multi-agent system is working properly while we need to abstract behavioural features from agents interaction for further analysis. Although work like [10] is closer to part of our work that they obtain agents' behavioural features by building a context model from agent interaction protocols, its purpose is still different from us. [10] is presenting a novel mechanism based on agent communication languages and interaction protocols for describing agents' behavioural features, but we focus on proposing an agent-based computing framework which aims to explore the relationship between their behavioural and non-behavioural patterns.

Regarding the analysis of the relationship between human behavioural features and MDD conditions, [11] exploits users' actions on social media to build an MDD prediction model, [2] explores the connection between depression and non-behavioural information. These contributions reveal several insights regarding characteristics of people with MDD, which can give inspiration to the analysis part of our work. But they do not take advantage of social norms appeared in interactions to analyse participants' behaviours. Differently, our system takes advantages of the norm specifications to analyse people's behavioural patterns. Some works try to analyse people's behavioural traits in ultimatum game which is carried out in our case study: [5] evaluates MDD's impact on decision-making in the ultimatum game, [6] explore people's genomic features variants on decision making in the ultimatum game. These studies have a weakness that once the researchers need to analyse new behavioural traits, they have to rebuild whole experiment system. Our method will give a practical, efficient, end-to-end solution for researchers to design behaviour experiment and analysis between samples' behavioural and non-behavioural patterns. The following sections present more details about ABIBA system and behaviour experiment.

3 ABIBA(Agent-Based Interaction Behaviour Analysis) system

3.1 System Computing Framework

ABIBA system is an automated, end-to-end, agent-based behaviour analysis system. It provides tools for the designers to make experiment protocols which describe agents' behaviour rules. The system will create several agents following the designers' protocol. Then the human players will control some of the agents to interact with other agents which are controlled by the system. The system leverages statistical analysis techniques like principal component analysis, correlation analysis and co-training methods to extract valuable information from collected interaction behaviours and classify agents into different clusters. Besides, the system can use data mining techniques to describe relationships between agents' behavioural patterns and non-behavioural traits like genomic information. Although in the case study, the system applies the analysis on the agents controlled by human players, the targeted agents can also be other entities like machines in industrial systems.
Fig.1 shows the system's working framework. There are two tasks the system will complete: sample analysis (shown in the bottom left light blue square) and population application. Sample analysis includes three steps: interaction data analysis, non-behavioural data analysis and relationship analysis. Interaction data analysis consists of interaction data collection and interaction data analysis. The two parts are carried out automatically by the interaction analysis subsystem. The system users will design interaction protocols which the system follows to maintain multi-agent interactions. The protocols define agents' behavioural rules like the

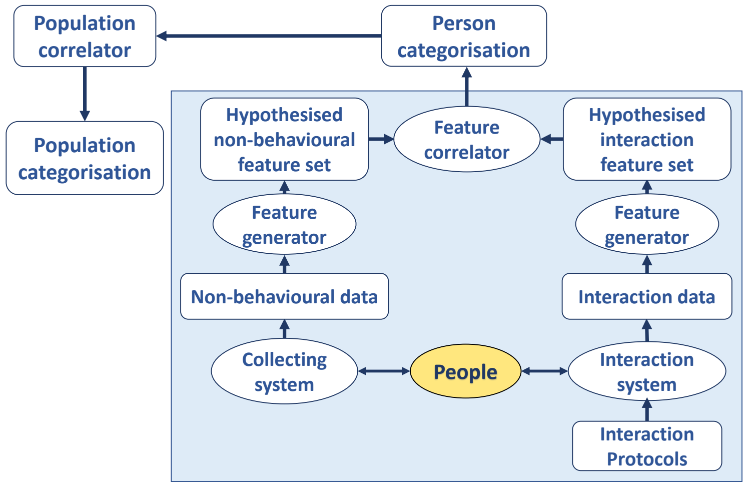

Fig. 1: ABIBA System Working Framework

agents' actions, actions' triggering constraints and agents' state transformation condition. Section 3.2 and 4.3 gives more details about the two concepts. Then the system follows the protocols to create multiple agents who follow the protocols to interact with each other. Finally, the interaction analysis subsystem will store agents' actions, extract their behaviour features for categorising them into subsets. Regarding non-behavioural patterns analysis, the system reads interaction players' non-behavioural patterns and extract typical features from these patterns. The system will explore the relationship between the two kinds of patterns like building a statistical classifier. The next section will explain the essential concepts in LCC.

3.2 Lightweight Coordination Calculus

Lightweight Coordination Calculus (LCC) [8] [4] is a comparatively simple but flexible, practical, executable specification language. We use it to design interaction protocols specifying multi-agent interaction. LCC is the first process calculi defining social norm used directly in the computation of multi-agent system [7].
To let readers better some critical concepts of LCC calculus used in ABIBA, we need to firstly sketch a framework for describing the multi-agent interaction in our system. The multi-agent interactions are presented as dialogic activities, called "scenes", involving different groups of agents playing various roles. The roles' descriptions are made in electronic institutions which define the roles' identities, their behavioural rules in the form of interaction protocols. In each activity, each agent follows its role's protocol to interact with others. Before agent makes any action, it will check if the action is allowed by their behavioural rules. We use LCC as specification description tool to design the protocols. If the readers want to explore more details about LCC [7] [8] will be helpful by giving comprehensive explanations about LCC syntax,

computing framework and application examples. An example of LCC implementation is shown in Fig 2. It's a part of the protocol used in our case study experiment which defines a role “proposer”:

```
a(proposer(Total),P)::=
    offer(X)=>a(responder(Total),R)<--e(offernum(X, R))
    then
    decide(D,X)<=a(responder(Total),R)
    then
    k(fair(D,X,Total,R)).
```

Fig. 2: LCC implementation of the proposer in ultimatum Game Protocol

The protocol presents two sorts of information:

1. Role's information: LCC protocol uses “a(N(V),I)” to define role's information, “N” is the role's name, “V” is the role's trait and “I” is the role's identity. Fig 2 uses “a(proposer(Total),P)” to define a role with name(“proposer”), trait(“Total”) and an identity(“P”).
2. Role's interaction actions: LCC protocol defines the role's actions in a form of exchanging messages: ‘M(X)=>a(N(V),I)<-e(Y)”. “M(X)” is the message, “M” is its name, “X” is its content. “=>” means sending a message and “<=” stands for receiving a message. Sometimes the role has to satisfy a constraint to make an action. The constraint is “e(Y)” following “<-”. In Fig 2, the role will firstly send a message when the constraint “e(offernum(X, R))” is satisfied. “then” is a state connector which defines the relationship between the actions. In Fig 2, the proposer will get the message, “decide(D, X)” from the responder after sending the offer message. Sometimes a role obtains new knowledge, “k(L(X))”, from the interaction. In Fig 2, the role gets knowledge “k(fair(D,X,Total,R))” after it receives the reply message.

3.3 Interaction Analysis

Interaction environment Once the system user complete the interaction protocol, the system follows them to build an environment for multiple agents to interact. Fig 3 shows the structure of the interaction environment. The human players can join the interactions through their PC or mobile phones. We call data generated from the human-system interaction as “interaction data”. The system transforms this kind of data into the “experimental data” for the system's interaction engine. The interaction engine is responsible for maintaining multi-agents interactions.

Interaction engine We have constructed an interaction engine based on LSCitter [4] inside the ABIBA system to manage multi-agent interactions. The engine follows the interaction protocol to create multiple virtual agents in a digital environment. Some of these agents, called

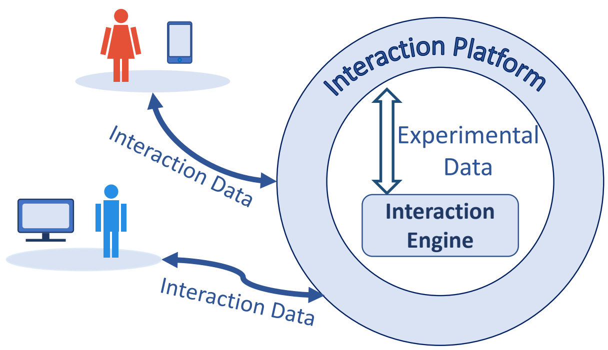

Fig. 3: Interaction Environment

"representative agents", are controlled by the entities from outside like human players in the case study. Whenever a virtual agent sends a message, the interaction engine will check the behavioural rule defined in the interaction protocols. In the case, the interaction engine will contact the human players about their choices. Once the engine receives the player's reply, the representative agents send the selected message. This is shown in Fig 4.

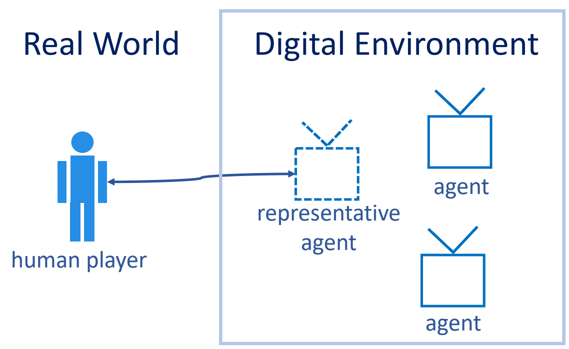

Fig. 4: The digital environment and the outside environment

The interaction engine has four parts: an agent engine, a protocol processor, a group communication unit and a server. Fig 5 shows the structure. The agent engine is responsible for creating agents. There are following elements inside the engine:

1. A state rewriting engine which changes agent state as the interaction progresses.
2. An agent state driver which keeps track of where the agents are in any given interaction.
3. A constraint satisfaction engine which brings facts and knowledge into the interactions, base on which the rewriting engine can rewrite the agent's state.
4. A knowledge storage engine which allows the agent to store knowledge. It is mostly the same as the satisfaction engine but may be different particularly if different kinds of satisfaction and storage are used, and the precedence is different.
5. A communication database which stores the agent's sent and received messages in an interaction.

The protocol processor reads the interaction protocol files, then parses the files into the protocols which describe all roles' information and

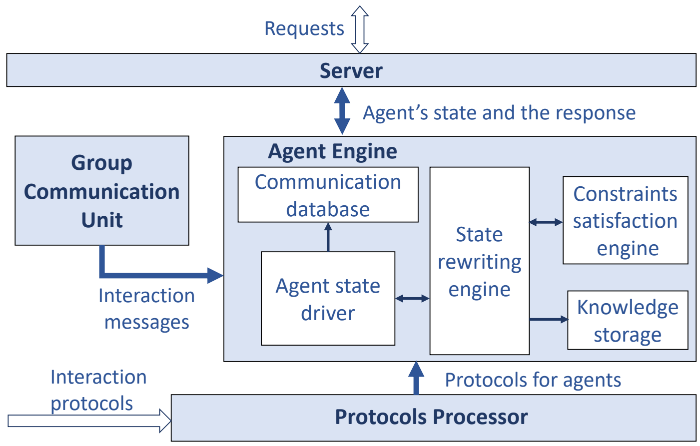

Fig. 5: Interaction Engine Structure

their actions. The agents will follow the protocol to communicate with each other. The group communication unit offers communication service which agent relies on to send and receive messages with each other.
The server in the interaction engine supports the communication between the engine and the outside environment. Take the case study as an example, each time a representative has to send a message, the agent engine will follow the protocols to send a request to the server about the human player's choice. Then the server contacts a web browser used by the player. When the player makes a response like submitting a number, the browser replies interaction engine and waits for the new request from the engine. Then the server reads the reply and sends a response to the agent engine. Finally, the representative agent sends the message with the input information.

Interaction behaviour analysis The system will extract behavioural features from the interaction behaviour, and then, categorise human samples based on their behavioural traits like their choices during the interactions. For example, the researcher wants to know people's preferences when they face a specific problem. The system will firstly start an experiment in which the human players have to make the targeted decisions. The system will collect these inputs and create features from them. Then the human players will be separated into different groups by their choices.

3.4 Relationship Analysis

After the interaction analysis, the system collects human participants' non-behavioural data like their genomic information to build new feature sets. Based on the interaction analysis results, the system will explore the relationship between human participants' behavioural and non-behavioural traits. The aim of the relationship analysis is to build a behaviour prediction model with non-behavioural data as input and behavioural patterns as output. In the case study, the system clusters samples into different groups by extracting their typical behavioural patterns from their actions in the experiments; then it uses the behavioural patterns as labels to build a classifier with samples' non-behavioural information (genomic data in this case) as features. The classifier reads people's non-behavioural data and predicts their potential behaviour patterns without additional interaction experiments. In the next section, we will show more details about the case study including experimental participants, experimental process and experimental instruments.

4 Case Study

4.1 Experimental participants

In collaboration with the MRC Institution of Genetics and Molecular Medicine (IGMM) in the College of Medicine Veterinary and Medicine who aims to break new ground in understanding the aetiology of MDD, the ABIBA system will support a behaviour experiment for the Generation Scotland cohort study(N=21000).

4.2 Experimental process

The ABIBA system will follow the protocols designed by the researchers and interact with the human participants. Then the system will analyse the relationship between participants' behavioural and non-behavioural patterns, consequently, provides the results to the researchers.
In the experiment, the researcher will firstly inform the human players about the game rules. Then the players choose their roles and exchange messages with other agents through the interfaces. There are two games for the players to play: the ultimatum game and the trust game.

The ultimatum game rule There are two roles in the ultimatum game: a proposer and a responder. At the beginning of the game, the proposer is given an amount of money. Then the proposer offers a part from the given money to the responder and waits for the reply. Finally, the responder accepts or rejects the offer. If the responder accepts the offer, it gets the amount of money, and the proposer gets the rest; if not, they will get nothing. For example, the proposer was given £10 and offers £5 to the responder. If the responder chooses to accept it, the responder gets £5, and the proposer gets the left £5; if the responder rejects the offer, they get nothing.

The trust game rule There are also two players in the trust game: an investor and a trustee. Firstly, the investor receives some money like £10. Then, the investor gives some amount to the trustee, say £4. Then the offer will be multiplied by a factor, like 3. The trustee will get the tripled offer, £12 in this case. Finally, the trustee repays a part of £12 to the investor, such as £3. Consequently, the investor gets £9 ($9 = 10 - 4 + 3$) and the trustee gets £9 ($9 = 4 * 3 - 3$).

4.3 Experimental instrument

Game interface Human participants will use the Generation Scotland website to as interface to play the game. Fig 6 shows how the interaction works: At first, the agent engine sends a state request to the server inside the interaction engine. The request is about the amount the "proposer's offer". Then the server sends a request to the web browser. The web page presents "How much do you want to offer?" and waits for the human player to enter their answer. The player enters the number into the text box and submits it. The web browser sends a request to the interaction engine. Then the server translates the request into an understandable message for the agent engine. After reading the message, the agent engine drives the proposer agent to send a message to the responder agent. When the interaction engine gets the reply from the responder agent, it shows the reply on the web page, in Fig 6 the reply is "The responder rejects the offer".

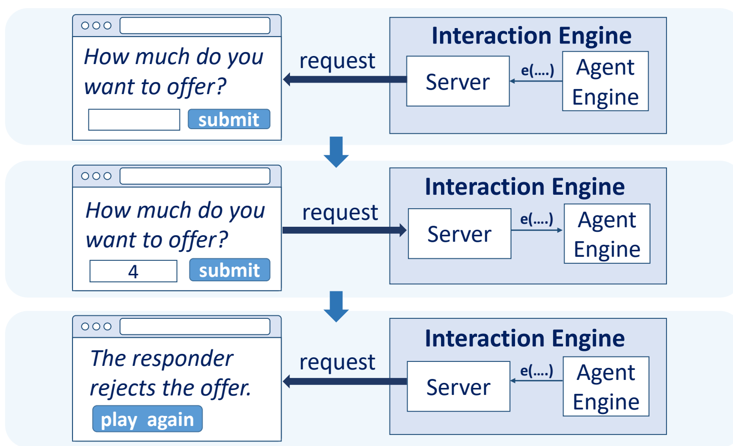

Fig. 6: Interactions in the ultimatum game

Game protocols The ABIBA system will follow the game protocol to organise the experiment. Section 3.2 has given an example of the protocol written by LCC. In the following part of this section, we will introduce the protocols used in the case study. Fig 7 and Fig 8 are the two protocols:
The ultimatum game protocol. The protocol is designed on the basis of the game rule mentioned in section 4.2. It contains two parts: the proposer part and the responder part. The proposer part has been explained in section 3.2, so we will only interpret the responder part in the below paragraph :

1. "a(responder(Total),R)": the role' name is "responder", its identity is "R", its characteristic is "Total" which means the amount of money given to the proposer at the beginning of the game.
2. "offer(X) <= a(proposer(Total),P)": the responder will wait for the message from the proposer. "X" stands for the amount of offer.
3. "then": it is a state connector means that the role should do the below action once the above one is done.

4. "k(get(Wi))< −−i(Wi is X*Rate)" means that once the responder receives the message about offer, it will compute the amount it will own and store the knowledge. "i(Wi is X*Rate)" stands for the computation action and "k(get(Wi))" stands for the storing action. "i()" is a symbol standing for computing action and "is" means assignment values from the right side to the left.
5. "decide(D,X) => a(proposer(Total),P) < −− e(acceptornot(D, X))": after the responder finishes the above actions, it reply to the proposer. When the responder satisfies the constraint "e(acceptornot(D, X))" by find true value for decision variable "D", it will send its decision message, "decide(D,X)" , to the proposer.

```
a(proposer(Total),P)::=
offer(X)=>a(responder(Total),R)<--e(offernum(X, R))
then
decide(D,X)<=a(responder(Total),R)
then
k(fair(D,X,Total,R)).

a(responder(Total),R)::=
offer(X)<=a(proposer(Total),P)
then
decide(D,X)=>a(proposer(Total),P)<--e(acceptornot(D, X)).
```

Fig. 7: The ultimatum game protocol

The trust game protocol is shown in Fig 8. In this protocol, there are also two roles, five actions and two constraints: the investor part: "a(investor(Total,Rate),I)" and "a(trustee(Rate),T)" define two the roles: an "investor" and a "trustee". "I" and "T" stand for identities of two roles. "Total" is the amount the investor gets at the beginning of the game. "Rate" is the times the investor's offer will be multiplied.

1. the investor part:
   (a) "offer(X) => a(trustee(Rate), T) < −− e(invest(X, T))": the investor will offer some money to the trustee when the constraint,"e(invest(X,T))", is satisfied. "X" stands for the number.
   (b) "repay(Y)<=a(trustee(Rate),T)": the investor waits for the reply form the trustee after sending the offer.
   (c) "k(own(Pi)) <– i(Pi is Total+(Y-X))": when the investor agent receives the repay message from the "trustee" agent, it will calculate the amount it finally has and store the amount.
2. the trustee part:
   (a) "offer(X) <= a(investor(_,Rate),I)": the trustee waits for the message from the investor.
   (b) "k(get(Wi))<–i(Wi is X*Rate)": when it gets the message, it will calculate the total amount it gets, "i(Wi is X*Rate)".

(c) “repay(Y) => a(investor(_,Rate),I) < −− e(repay(Y, I))”: once constraint, “e(repay(Y,I))”, is satisfied, the trustee will repay to the investor by sending “repay(Y)”. “Y” stands for the repay amount.

(d) “k(own(Pt)) <– i(Pt is Wi-Y)”: after repaying the offer, the trustee will calculate the remaining amount and store the amount.

```
a(investor(Total,Rate), I)::=
offer(X)=>a(trustee(Rate), T)<--e(invest(X, T))
then
repay(Y)<=a(trustee(Rate),T)
then
k(own(Pi))<--i(Pi is Total+(Y-X)).

a(trustee(Rate),T)::=
offer(X)<=a(investor(_,Rate),I)
then
k(get(Wi))<--i(Wi is X*Rate)
then
repay(Y)=>a(investor(_,Rate),I)<--e(repay(Y, I))
then
k(own(Pt))<--i(Pt is Wi-Y).
```

Fig. 8: The trust game protocol

5 Following Work

In summary, he ABIBA system takes advantage of specification language LCC to offer an end-to-end solution for agent interaction behaviour analysis. It exploits social norm theory to build an agent interaction model for organising multiple player experiments. To test the system function and performance, we apply ABIBA to a case study about behaviour and biologic patterns analysis among people with MDD in cooperation with the IGMM and Generation Scotland. We will collect data starting with 200 people for a pilot study to be sure things run smoothly. We anticipate publishing the specific results of the behavioural experiments in a future paper since these will require detailed analysis concerning genotypic information.

References

1. Abushark, Y., Thangarajah, J., Miller, T., Harland, J., Winikoff, M.: Early Detection of Design Faults Relative to Requirement Specifications in Agent-Based Models. In: Proceedings of the 2015 In-

ternational Conference on Autonomous Agents and Multiagent Systems. pp. 1071–1079. AAMAS '15, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC (2015)
2. Kessler, R.C., van Loo, H.M., Wardenaar, K.J., Bossarte, R.M., Brenner, L.A., Cai, T., Ebert, D.D., Hwang, I., Li, J., de Jonge, P., Nierenberg, A.A., Petukhova, M.V., Rosellini, A.J., Sampson, N.A., Schoevers, R.A., Wilcox, M.A., Zaslavsky, A.M.: Testing a machine-learning algorithm to predict the persistence and severity of major depressive disorder from baseline self-reports. Molecular Psychiatry 21(10), 1366–1371 (2016)
3. Lomuscio, A., Qu, H., Raimondi, F.: MCMAS: an open-source model checker for the verification of multi-agent systems. International Journal on Software Tools for Technology Transfer 19(1), 9–30 (2017)
4. Murray-Rust, D., Robertson, D.: LSCitter: Building Social Machines by Augmenting Existing Social Networks with Interaction Models (2014)
5. Radke, S., Schäfer, I.C., Müller, B.W., de Bruijn, E.R.: Do different fairness contexts and facial emotions motivate 'irrational' social decision-making in major depression? An exploratory patient study. Psychiatry Research 210(2), 438–443 (2013)
6. Reuter, M., Felten, A., Penz, S., Mainzer, A., Markett, S., Montag, C.: The influence of dopaminergic gene variants on decision making in the ultimatum game. Frontiers in human neuroscience 7(June), 242 (2013)
7. Robertson, D.: A Lightweight Coordination Calculus for Agent Systems, pp. 183–197. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)
8. Robertson, D.: Lightweight coordination calculus for agent systems: Retrospective and prospective. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). vol. 7169 LNAI, pp. 84–89 (2012)
9. Serrano, E., Muñoz, A., Botia, J.: An approach to debug interactions in multi-agent system software tests. Information Sciences 205, 38–57 (2012)
10. Serrano, E., Rovatsos, M., Botía, J.A.: Data mining agent conversations: A qualitative approach to multiagent systems analysis. Information Sciences 230, 132–146 (2013)
11. Shickel, B., Heesacker, M., Benton, S., Rashidi, P.: Hashtag Healthcare: From Tweets to Mental Health Journals Using Deep Transfer Learning pp. 1–10 (2017)
12. The National Institute for Health and Care Excellence: Depression: The NICE Guideline on the Management of Depression in Adults (2009)